Composite Retrieval of Diverse and Complementary Bundles

Sihem Amer-Yahia, Francesco Bonchi, Carlos Castillo, Esteban Feuerstein, Isabel Mendez-Diaz, and Paula Zabala

Abstract—Users are often faced with the problem of finding complementary items that together achieve a single common goal (e.g., a starter kit for a novice astronomer, a collection of question/answers related to low-carb nutrition, a set of places to visit on holidays). In this paper, we argue that for some application scenarios returning *item bundles* is more appropriate than ranked lists. Thus we define *composite retrieval* as the problem of finding k bundles of complementary items. Beyond complementarity of items, the bundles must be valid w.r.t. a given budget, and the answer set of k bundles must exhibit diversity. We formally define the problem and show that in its general form is NP-hard and that also the special cases in which each bundle is formed by only one item, or only one bundle is sought, are hard. Our characterization however suggests how to adopt a two-phase approach (*Produce-and-Choose*, or PAC) in which we first produce many valid bundles, and then we choose k among them. For the first phase we devise two ad-hoc clustering algorithms, while for the second phase we adapt heuristics with approximation guarantees for a related problem. We also devise another approach which is based on first finding a k-clustering and then selecting a valid bundle from each of the produced clusters (*Cluster-and-Pick*, or CAP). We compare experimentally the proposed methods on two real-world data sets: the first data set is given by a sample of touristic attractions in 10 large European cities, while the second is a large database of user-generated restaurant reviews from Yahoo! Local. Our experiments show that when diversity is highly important, CAP is the best option, while when diversity is less important, a PAC approach constructing bundles around randomly chosen pivots, is better.

Index Terms—Composite retrieval, maximum edge subgraph, complementarity, diversity

1 Introduction

Online search has become a daily activity and a source of a variety of valuable information, from the finest granularity such as finding the address of a specific restaurant, to more complex tasks like looking for accessories compatible with an iPhone or planning a trip. The latter typically involves running multiple search queries to gather information about different places, reading online reviews to find out about hotels, and checking geographic proximity of places to visit. We refer to this information seeking activity as *composite retrieval* and propose to organize results into *item bundles* that together constitute an improved exploratory experience over ranked lists.

As a first step towards composite retrieval definition, we need to formalize intuitive desirable properties of item bundles. We distinguish between properties of each bundle in the answer and properties of the answer as a whole. Given the wide applicability of composite retrieval, we propose to first explore these properties in a few application scenarios, then we motivate the need for a framework to study composite retrieval complexity and develop efficient algorithms.

1.1 Composite Retrieval Scenarios

Consider the case of a user selecting the restaurants to try during a visit to a new city. In this scenario, the user has a limited budget which might be either financial, or simply the number of nights to spend in the city. Moreover the user prefers suggested restaurants to serve different cuisines. In this setting the validity of a bundle of restaurants is given by the budget constraint and the complementarity of the restaurants in the bundle w.r.t. the cuisine they serve. Other restaurant attributes could be used for defining valid bundles. For example, instead of cuisines, different dress codes (e.g., casual, business casual, formal) could distinguish restaurants in a single bundle.

Moreover, in order to provide meaningful bundles, restaurants forming each bundle must be compatible, e.g., close geographically, or liked by similar reviewers. The degree of compatibility of the items forming a bundle defines the quality of the bundle. Intuitively, in the case geographic distance is used, the closer restaurants are from each other, the higher the quality of the bundle they belong to. Similarly, when common reviewers are used as the quality of a bundle, the higher the overlap in similar reviewers within the same bundle, the higher the quality of that bundle. Finally, bundles forming an answer set can be generated to cover different various geographic areas, or different reviewers segments, thereby producing an answer set of bundles with diversity.

- *S. Amer-Yahia is with the CNRS - LIG, Grenoble, France. E-mail: Sihem.Amer-Yahia@imag.fr.*
- *F. Bonchi is with the Yahoo Labs, Avinguda Diagonal 177, 8th floor, Barcelona, Catalunya, Spain. E-mail: bonchi@yahoo-inc.com.*
- *C. Castillo is with the Qatar Computing Research Institute, Doha. E-mail: chato@acm.org.*
- *E. Feuerstein, I. Mendez-Diaz, and P. Zabala are with the Department of Computer Science, Universidad de Buenos Aires, Argentina. E-mail: {efeuerst, imendezg, pzabala}@dc.uba.ar.*

Manuscript received 3 Mar. 2013; revised 14 Jan. 2014; accepted 28 Jan. 2014. Date of publication 16 Feb. 2014; date of current version 26 Sept. 2014. Recommended for acceptance by J. Pei.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2014.2306678

In general, browsing and searching social media, e.g., a photo-sharing site, or a question-answering portal, might benefit from composite retrieval. Given a query or while browsing a category, diverse bundles of complementary resources might provide a better experience for the user. Complementarity of resources forming a bundle can be expressed, e.g., using topical sub-categories in order to ensure that no two resources in the same bundle are on the same sub-category. Budget can simply be the number of resources forming a bundle. Compatibility of two resources can be expressed by their similarity, e.g., the Jaccard coefficient of their tags, or the overlap of the users that expressed interest for the resources—thereby bundling together resources from the same user community. In this last case, diversity is achieved by finding multiple bundles each of which representing the opinion of a different user population.

Other applications include online shopping where products forming the same bundle have different types (e.g., a telescope, a carrying bag, a lens kit, an astronomy book), their total price is within budget, and they are compatible according to their manufacturers. City tours can also be represented as a bundle with one item per attraction (e.g., museum, monument, market, park), budget is the total time required to visit all items in a bundle, or the total cost of the entrance tickets, and compatibility is an inverse of geographic distance. Here, diversity means attractions in different areas of the city. Finally, team building for problem solving is another application scenarios for composite retrieval. Here we want to find experts on a topic (e.g., safety standards experts) to form a team. The team needs to contain complementary members with different roles—e.g., a manager, an engineer, a lawyer—with an upper limit on the team size. Compatibility requires experts to have worked together in the past.

1.2 Article Contributions and Roadmap

Our work lays the theoretical foundations for composite retrieval, a general framework under which different variants of the problem can be defined by constraining some dimensions and optimizing others.

The above examples show the wide applicability of composite retrieval beyond traditional information retrieval and the variety of complementarity and budget constraints that could be used. It is important to note that bundles may be built using the most relevant items to a query thereby making traditional relevance orthogonal to bundle construction. That allows us to define the quality of a bundle, i.e., its score, as a function of pair-wise similarities between its items. As in traditional retrieval, we aim to retrieve highly scoring and also *diverse* bundles. The quality of a collection of k bundles is given by a weighted combination of the quality of each bundle and inter-bundle diversity. While we provide a proof of concept with two different data sets, for specific future applications user studies would need to be deployed.

Our contributions can be summarized as follows:

- We define and study the problem of retrieving k diverse bundles of complementary items under a per-bundle budget. In Section 2 we formally define the problem and we provide two different proofs of **NP**-hardness, highlighting two different aspects of the complexity of the problem. Both proofs reduce the well known **NP**-complete MAXIMUM EDGE SUBGRAPH problem to ours.
- Given that our problem is **NP**-hard, we turn our attention to approximation algorithms. Following the hint of our **NP**-hardness proofs, in Section 3 we describe a two-phase approach (*Produce-and-Choose*, or PAC) in which we first produce many valid bundles, and then we choose k among them. For the choosing phase we show an approximation-preserving reduction from MAXIMUM EDGE SUBGRAPH which enables us to adopt heuristics that have been developed in the literature for that problem.
- For the task of producing good bundles we observe the similarity between the objective function of our problem, and that of clustering. Following this observation, in Section 4 we devise two ad-hoc clustering algorithms: the first one based on constrained hierarchical clustering, and the second one inspired by *k-nn* clustering. In Section 5 we introduce a different approach which is based on first finding a k-clustering and then selecting a valid bundle from each of the produced clusters and in Section 6 we present an integer programming (IP) formulation of our problem, which may be used to obtain the exact solution.
- In Section 7 we compare experimentally the proposed methods on two real-world data sets. The first is given by a sample of 20 touristic attractions in 10 large European cities, while the second is a large database of user-generated restaurant reviews from Yahoo! Local. We assess both quality and efficiency. We show that our heuristics produce good results according to our optimization objective, comparing them with the results of the IP implementation within a Branch-and-Cut framework. Our main finding is that the performance of these methods depends basically on a parameter controlling the tradeoff between the average score of the bundles and the diversity of the set of bundles. When diversity is highly important, we obtained the best performance using algorithms based on creating a global clustering of the items first, and then choosing bundles that respect those clustering boundaries. When diversity is less important, we show that "local" methods that construct good bundles around randomly chosen pivots produce better results.

2 Problem Definition and Complexity

We are given a set of items $\mathcal{I}$. Each item in $\mathcal{I}$ is uniquely identified and has a set of attributes. We assume a *similarity* value $s(u, v)$ for each pair of items $(u, v) \in \mathcal{I} \times \mathcal{I}$. The similarity values $s(u, v)$ may be provided explicitly in the input, or computed implicitly from the representation of the items. For instance, if items are documents, $s(u, v)$ may be defined as the cosine between the vectors that represent the documents u and v; if items are restaurants, $s(u, v)$ may be defined as the fraction of reviewers that like both u and v; etc. Hereinafter, we simply consider the values $s(u, v)$ as

input to our problem and we do not make any assumption on how to obtain those values.

Sometimes it is useful to think of our input as a complete, undirected and weighted graph $G = (\mathcal{I}, E, s)$, where each edge (u, v) has a weight $s(u, v)$. For sake of clarity of presentation, we consider that the similarity function s takes values in the interval $[0, 1]$ (this is always achievable by normalization).

For convenience we also refer to the *distance function* between item pairs, defined as $d(u, v) = 1 - s(u, v)$, that also takes values in the interval $[0, 1]$ (although by definition we cannot warrant that $d(u, v)$ defined this way is really a metric, i.e., it obeys triangular inequality).

Our goal is to retrieve a set of bundles $\mathcal{S} = \{S_1, \ldots, S_k\}$, where a bundle $S_i \in 2^{\mathcal{I}}$ is a set of items that satisfy constraints of *complementarity* and *budget* as expressed in the following definition.

Definition 1 (Valid Bundle). *Given a set of items $\mathcal{I} = \{i_1, \ldots, i_n\}$, a bundle $S \in 2^{\mathcal{I}}$ is said to be* valid *iff it satisfies the following two constraints:*

- *Complementarity. Given a property α of the items (e.g., an attribute), no two items in $S_i \in \mathcal{S}$ exhibit the same value for that property: i.e., $\forall u, v \in S_i, u.\alpha \neq v.\alpha$.*
- *Budget. Given a set-valued non-negative and monotone function $f : 2^V \rightarrow \mathbb{R}^+$, and given a budget threshold β, we require that $\forall S_i \in \mathcal{S}, f(S_i) \leq \beta$. Typical examples of budget are simply the number of items forming a bundle or an upper-bound on the sum of the costs of items forming the bundle, given a cost attribute.*

For example, in the case of restaurants, the property α could be cuisine type (e.g., Chinese, American) or geographic area (e.g., East Village, Greenwich Village, Lower East side) or a combination thereof.

The budget function f could be the sum of the average price of a meal at all restaurants forming the bundle. In a travel application, items are specific attractions, α their type (e.g., Museum, Park), and f can be the time required to visit each place.

We are now ready to define the problem of composite retrieval.

Problem 1 (Composite Retrieval). *Given a set of items $\mathcal{I} = \{i_1, \ldots, i_n\}$, a pair-wise similarity function $s(u, v)$ for each $(u, v) \in \mathcal{I} \times \mathcal{I}$, a complementarity attribute α, a budget function $f : 2^{\mathcal{I}} \rightarrow \mathbb{R}^+$, a budget threshold β, and an integer k, find a set $\mathcal{S} = \{S_1, \ldots, S_k\}$ of valid bundles that maximizes:*

$$\sum_{1 \leq i \leq k} \sum_{u,v \in S_i} \gamma \, s(u, v) + \sum_{1 \leq i < j \leq k} (1 - \gamma)(1 - \max_{u \in S_i, v \in S_j} s(u, v)),$$

where γ is a user-defined scaling parameter.

The objective function resembles a typical clustering objective, where the total quality of the clustering is expressed as a weighted combination of the quality of single clusters (which in turn is defined as their intra-cluster cohesion) and inter-cluster separation. The latter can be defined as the minimum distance between an item in one bundle and an item in another one. Intra-cluster cohesion reflects cluster quality as a function of the similarity or cohesion between items forming the cluster. Inter-cluster separation reflects answer diversity.

Unlike standard clustering, our problem does not seek a total partitioning of items, instead it aims at finding k good groups, that might potentially be small as they are bounded by the budget constraint. Therefore, some items in $\mathcal{I}$ might not belong to any bundle or belong to more than one. It is worth noting that our problem definition, by summing over all elements in a bundle, favors larger bundles: as large as possible given the budget constraint.

The complementarity requirement, requiring no more than one single element of a given kind to belong to a bundle, can be seen as a set of many *cannot-link* constraints typical to constrained clustering [5]. In particular, given the complementarity property α, each item *cannot-link* with all the other items in $\mathcal{I}$ that have the same value for α. These observations will be used later in Section 4 to devise algorithms for composite retrieval.

Finally, it is important to note that bundles are built using the most relevant items to a query thereby making traditional relevance orthogonal to bundle construction. That allows us to define the quality of a bundle, i.e., its score, as a function of pair-wise compatibilities between its items. Similarly to traditional retrieval, we aim to retrieve highly scoring and also *diverse* bundles. The quality of a collection of k bundles is given by a weighted combination of the quality of each bundle and inter-bundle diversity.

2.1 Problem Complexity

Not surprisingly our problem is hard. We provide two different proofs of **NP**-hardness, where the first one highlights the complexity of the first argument of the objective function, and the second one the complexity of the second argument. Both proofs reduce the well known **NP**-complete problem MAXIMUM EDGE SUBGRAPH (also known as *Dense k-subgraph*) to our problem. The MAXIMUM EDGE SUBGRAPH problem requires to find a set of m nodes, such that the induced subgraph has maximum sum of edge weights.

Theorem 1. *Composite retrieval is* **NP***-hard.*

Proof. We prove the hardness by reducing MAXIMUM EDGE SUBGRAPH to our problem. Given an instance $\mathcal{L}$ of MAXIMUM EDGE SUBGRAPH, consisting of a graph $G = (V, E)$, a weight function $w : E \rightarrow \mathbb{N}$, and an integer $m \leq |V|$, we create an instance $\mathcal{J}$ of composite retrieval as follows. For each node $v \in V$ we create an item $v \in \mathcal{I}$. Moreover we give the same value of complementarity attribute α to all items $v \in \mathcal{I}$, so that no bundle of size larger than one qualifies. Similarly we set the budget function to be the cardinality of a bundle and the budget threshold to be 1. The required number k of bundles is m. Finally, for each pair of items $u, v \in \mathcal{I}$, we set $s(u, v) = 1 - w(u, v)$ if $(u, v) \in E$, and $s(u, v) = 1$ otherwise. The reduction can clearly be carried out in polynomial time. Under that reduction, the left summation in the objective function of composite retrieval is null, as each bundle is formed only by one item. Therefore, the objective becomes to maximize

$$(1 - \gamma) \sum_{i \leq j \leq k} (1 - \max_{u \in S_i, v \in S_j} s(u, v)).$$

The term $(1 - \gamma)$ can be removed. Moreover we can identify each bundle with the unique element forming it.

Thus, we can rewrite the objective function as

$$\sum_{i \le j \le m} w(S_i, S_j).$$

Therefore a set $\mathcal{S} = \{S_1, \ldots, S_k\} \subseteq V$ is a solution in instance $\mathcal{L}$ of MAXIMUM EDGE SUBGRAPH *iff* it is a solution in instance $\mathcal{J}$ of composite retrieval. □

While the proof above exploits a reduction where each bundle can contain at most a single item, the next proof uses a reduction where a unique bundle is sought.

Proof. Given an instance $\mathcal{L}$ of MAXIMUM EDGE SUBGRAPH, consisting of a graph $G = (V, E)$, a weight function $w : E \to \mathbb{N}$, and an integer $m \le |V|$, we create an instance $\mathcal{J}$ of composite retrieval as follows. For each node $v \in V$ we create an item $v \in \mathcal{I}$. The parameter k of our problem is set to 1, i.e., we ask for a single bundle. We now assign a *different* value of the complementarity attribute α to all items $v \in \mathcal{I}$, so that any two items can be part of the same bundle. Similarly to the previous proof, we set budget function to be the cardinality of a bundle, but now the budget threshold is set to be m, i.e., we ask for a single bundle of size m. Finally, for each pair of items $u, v \in \mathcal{I}$, we set $s(u, v) = w(u, v)$ if $(u, v) \in E$. Clearly, this reduction can be carried out in polynomial time.

With this reduction, the *right* summation in the objective function of composite retrieval is null, as there is just one bundle in the answer. The objective function to maximize is

$$\sum_{1 \le i \le 1} \sum_{u,v \in S_i} \gamma s(u, v) = \sum_{u,v \in S_1} \gamma s(u, v) = \sum_{u,v \in S_1} \gamma w(u, v),$$

which is exactly the same as maximizing the objective function of MAXIMUM EDGE SUBGRAPH. □

Given that our problem is **NP**-hard, we turn our attention to heuristics and approximation algorithms. Going in that direction, it becomes crucial to see what is known about the MAXIMUM EDGE SUBGRAPH that we used in our reduction. In fact, in both reductions the number of nodes and edges remain equal, as well as the edges' weights. Then we conclude that all negative approximability results for MAXIMUM EDGE SUBGRAPH apply to Composite Retrieval.

The bad news is that MAXIMUM EDGE SUBGRAPH cannot be approximated within constant factors unless **P=NP** [6]. Moreover, it is believed that it is hard to approximate within a polylogarithmic factor, and the same non-approximability results apply to our problem. We will however direct our effort to heuristics that try to use some known non-constant approximation-guaranteed algorithms. In the following, m denotes the parameter of the MAXIMUM EDGE SUBGRAPH problem. Asahiro et al. [4] propose a simple heuristic that consists in repeatedly removing a vertex with the minimum weighted-degree in the currently remaining graph, until exactly m vertices are left. The approximation ratio is $O(n/m)$. Feige et al. [12] and very recently, Bhaskara et al. [6] give more complex heuristics improving the approximation ratios up to $O(n^{1/4} + \epsilon)$. In the significant particular case in which the weights obey the triangular inequality, a 2-approximation exists [16]. Note that, even if the max-dense-subgraph (i.e., where no bound on the size of the subgraph is given) is polinomialy solvable [14], our double NP-hardness proof shows that even if we removed the budget or complementarity constraints the problem would still NP-hard, as the complexity would be attached to the sub-problem of finding the most diverse set of k bundles.

Also note that, as it may be seen from the above **NP**-hardness proofs, even the particular problem of finding just one (the best) bundle is hard, whenever there are budget constraints. However, in some particular cases the problem can be solved in polynomial time. These are the cases in which the bundle size is bounded by some constant. That happens, for example, whenever the number of different values of the complementarity attribute α is bounded by a constant, so in $O(n^{|\alpha|})$ time we can solve the problem by considering all possible item combinations that fulfill the complementarity constraints. A similar approach can be taken when the cardinality of the bundles is limited by a constant upper-bound. In all these cases a brute force approach could be feasible.

3 PRODUCE BUNDLES AND CHOOSE

In the previous section, we showed the complexity of our problem by means of two different reductions from MAXIMUM EDGE SUBGRAPH. The first of the two reductions suggests a possible approach for composite retrieval. In fact, if we generate candidate bundles and we consider each candidate bundle as a node of a *bundle-graph*, where inter-bundle distances are the edge weights, then we are again in front of the MAXIMUM EDGE SUBGRAPH problem.

This suggests that composite retrieval can be solved by generating a set of candidate bundles and then selecting the best possible subset. We call this approach *Produce-and-Choose* (Algorithm 1).

Algorithm 1 `Produce − and − Choose`

Input: $\mathcal{I}$, α, f, β, k, γ as in Problem 1
Output: A set $\mathcal{S}$ of k valid bundles.
1: $Cand \leftarrow$ `produce_bundles`$(\mathcal{I}, \alpha, f, \beta)$;
2: $G \leftarrow$ `build_bundle_graph`$(Cand)$;
3: **return** `ChooseBundles`(k, γ, G);

The more bundles we generate in the Produce phase, the better the result we can expect to have. If the set of candidates $Cand$ contains all possible bundles we can obtain the exact solution (but we cannot do that polynomial time) or an approximate one (in polynomial time).

Thus we tackle our problem by generating proper subsets of candidate bundles and trying to smartly select among them.

In the rest of this section we focus on the Choose phase (algorithm `ChooseBundles`) and show that we can exploit the known results for MAXIMUM EDGE SUBGRAPH, discussed in the previous section, while preserving approximation guarantees. For this, we need to define an intermediate problem, in which both edges and nodes are weighted, and where we need to find a maximum weight (considering edges AND nodes) k-node induced subgraph.

Formally, the MAXIMUM EDGE-NODE SUBGRAPH problem is the following: the input consists in a graph $G = (V, E)$, a weight function $\psi : E \to \mathbb{N}$, a weight function $\omega : V \to \mathbb{N}$, an integer $k \leq |V|$ and a real value $\gamma \in [0, 1]$. The output is a set $V' \subseteq V$ such that $|V'| = k$ that maximizes the total weight of the edges and the nodes in the subgraph induced by V', denoted $G' = (V', E')$, weighted by the parameter γ, as follows:

$$\gamma \sum_{u \in V'} \omega(u) + (1 - \gamma) \sum_{(u,v) \in E'} \psi(u, v).$$

It is straightforward to see the direct relation between our PAC approach to composite retrieval (i.e., first produce bundles and then choose those maximizing the objective function), and the MAXIMUM EDGE-NODE SUBGRAPH problem defined above. In particular bundles are nodes, quality of a bundle (i.e., cohesion) is the node's weight, and inter-bundle distances are edge weights. We next show an approximation-preserving reduction from instances of MAXIMUM EDGE-NODE SUBGRAPH to MAXIMUM EDGE SUBGRAPH that will allow us to apply any of the approximated algorithms for the MAXIMUM EDGE SUBGRAPH problem and maintain the same approximation guarantee. Note that if *Cand* is not complete, we loose all approximation guarantees. Intuitively, however, the better approximation ratios we have for the Choose phase, the better results we can expect.

Consider an instance of MAXIMUM EDGE-NODE SUBGRAPH: $G = (V, E), \psi(u, v), \omega(u), k$ and γ. We transform it to an instance of the MAXIMUM EDGE SUBGRAPH where $G = (V, E)$ and k are the same, and the edges weight function is:

$$w(u, v) = \frac{\gamma}{2(k-1)}(\omega(u) + \omega(v)) + (1 - \gamma)\psi(u, v).$$

We can see that a solution V' (and its induced subgraph $G' = (V', E')$) maximizing the instance of MAXIMUM EDGE SUBGRAPH is also maximizing the corresponding instance of MAXIMUM EDGE-NODE SUBGRAPH:

$$\begin{aligned}
&\sum_{(u,v) \in E'} w(u, v) \\
&= \frac{\gamma}{2(k-1)} \sum_{(u,v) \in E'} (\omega(u) + \omega(v)) + (1 - \gamma) \sum_{(u,v) \in E'} \psi(u, v) \\
&= \frac{\gamma}{2(k-1)} ((k-1) \sum_{u \in V'} \omega(u) + (k-1) \sum_{v \in V'} \omega(v)) \\
&\quad + (1 - \gamma) \sum_{(u,v) \in E'} \psi(u, v) \\
&= \frac{\gamma}{2(k-1)} 2(k-1) \sum_{u \in V'} \omega(u) + (1 - \gamma) \sum_{(u,v) \in E'} \psi(u, v) \\
&= \gamma \sum_{u \in V'} \omega(u) + (1 - \gamma) \sum_{(u,v) \in E'} \psi(u, v).
\end{aligned}$$

The best approximation guarantee for MAXIMUM EDGE SUBGRAPH is given for the case that edge weights, i.e., inter-bundle distance, is a metric [16]. Unfortunately, in our problem where inter-bundle distance is defined as $(1 - \max_{u \in S_i, v \in S_j} s(u, v))$, or equivalently as the minimum distance $d(u, v) = 1 - s(u, v)$, triangular inequality does not hold. This can be seen through the following family of counter-examples, that can be present whenever bundles are of size greater than one. Consider three bundles A, B and C. There may be an item $b \in B$ that has maximum similarity with an item $a \in A$, and another item $b' \in B$ with maximum similarity with an item $c \in C$, while there does not exist an item in A with maximum similarity with any item in C. In this setting the distance between bundles A and C would be larger than zero, while distances between A and B, and between B and C would be both null, i.e., triangular inequality does not hold. Thus, we cannot borrow the two-approximation of [16], and will need to refer to some of the heuristics that have been proposed in [4], [6], [12]. The first of them may be applied to the weighted version of the problem, while the other two were developed for the unweighted version and can be extended to the weighted version incurring an additional $O(\log n)$ factor in the approximation ratio [12]. This is done by first scaling edge weights to n^2 and then applying a procedure that consists in dividing the edges into $2 \log n$ buckets according to their weights, then solving separately the $2 \log n$ instances of the unweighted problem obtained by including only the edges in each bucket, and finally selecting the best of the $2 \log n$ solutions obtained.

Among the three heuristics just mentioned, we choose to implement the one in [4], due to its simplicity and applicability to the weighted case.

Algorithm 2 `ChooseBundles`

Input: k, γ, and the bundle weighted graph $G = (V, E)$ where $\forall S \in V : \omega(S) = \sum_{u,v \in S} s(u, v)$, and $\forall (S_i, S_j) \in E : \psi(S_i, S_j) = 1 - \max_{u \in S_i, v \in S_j} s(u, v)$.
Output: A set $\mathcal{S}$ of k valid bundles.
1: Define $w(u, v) = \frac{\gamma}{2(k-1)}(\omega(u) + \omega(v)) + (1 - \gamma)\psi(u, v)$.
2: $\mathcal{S} \leftarrow V$
3: **while** $|\mathcal{S}| > k$ **do**
4: $\quad u \leftarrow \text{argmin}_{[u \in \mathcal{S}]} \sum_{v \in \mathcal{S}} w(u, v)$;
5: $\quad$ Remove u from $\mathcal{S}$
6: **return** $\mathcal{S}$

Algorithm 2 provides the pseudocode for the Choose phase of our method. It receives in input a set of valid bundles according to Definition 1. This set of bundles is represented as a complete weighted graph, where nodes are bundles. Nodes and edges are weighted with intra-bundle cohesion and inter-bundle separation respectively.

Given this graph and the parameters γ and k, this is an instance of MAXIMUM EDGE-NODE SUBGRAPH that can be educed to an instance of MAXIMUM EDGE SUBGRAPH by setting the edges' weight as in line 1. Then we can apply the heuristic of [4] greedily removing at each iteration the vertex with minimum weighted-degree (lines 4-5) in the currently remaining graph, until exactly k vertices are left (line 3).

4 Creating Good Bundles

In this section we devise methods for the `produce_bundles` phase of our method. As we discussed in Section 2, our objective function resembles a typical clustering objective function, where the total quality of the clustering is expressed as a weighted combination of the quality of single clusters and the inter-cluster separation. Additionally, complementarity within each bundle can be enforced by means of a set of *cannot-link* constraints.

Following these observation we propose two alternative algorithms for the task of producing a set of good valid bundles $Cand$, with given cardinality $|Cand| = c \gg k$. The first method is based on constrained hierarchical clustering [10], while the second takes inspiration from *k-nn* clustering.

4.1 Constrained Clustering

The first option we consider is to perform a constrained hierarchical agglomerative clustering (`C-HAC`, Algorithm 3) using $d(u,v) = 1 - s(u,v)$ as distance among items u and v. `C-HAC` starts with a number of clusters equal to the number of input elements, and then iteratively merges the closest clusters until a stop condition (e.g., number of clusters) is met. Two clusters S_1, S_2 such that the resulting bundle $S_1 \cup S_2$ is not valid, i.e., if it does not honor budget or complementarity constraints, cannot be merged.

Algorithm 3 `C-HAC`

Input: $\mathcal{I}$, α, f, β, and number of bundles c
Output: a set of c valid candidate bundles

1: $Cand \leftarrow \cup_{i \in \mathcal{I}} \{i\}$
2: **while** $|Cand| > c$ **do**
3: $bestscore \leftarrow -\infty$
4: $bestcandidate \leftarrow \emptyset$
5: **for** (each $S_i \in Cand$) **do**
6: **for** (each $S_j \in Cand; S_j \neq S_i$) **do**
7: **if** (`validMerge`$(S_i, S_j, \alpha, f, \beta)$) **then** // $S_i \cup S_j$ form a valid bundle
8: **if** ($score(S_i \cup S_j) > bestscore$) **then**
9: $bestscore \leftarrow score(S_i \cup S_j)$
10: $bestcandidate \leftarrow \{S_i, S_j\}$
11: **if** ($bestcandidate = \emptyset$) **then** // no more merging is possible
12: **break**
13: $Cand \leftarrow Cand - S \quad \forall S \in bestcandidate$
14: $Cand \leftarrow clusters \cup bestcandidate$
15: **return** $Cand$

The verification of whether two bundles can be merged into a valid bundle (`validMerge` in line 7) can be made faster by observing that if $S_1 \cup S_2$ is not a valid bundle, then $(S_1 \cup T) \cup (S_2 \cup V)$ is not a valid bundle either for any $T, V \subseteq \mathcal{I}$. This means that a graph of "incompatibility" (connecting clusters that cannot be merged) can be kept, and after every merge in the algorithm, this graph can be updated simply by lumping the nodes corresponding to the merged clusters and keeping all their non-redundant edges. Note that in this algorithm each cluster is always a valid bundle, so when it reaches the stopping condition $|Cand| = c$ we can immediately return $Cand$ without the need for any further check.

4.2 Bundles One-by-One (BOBO)

The second method for producing a good set of candidate bundles is inspired by *k-nn* clustering. At each step an item is chosen as pivot, and a valid bundle is built around that pivot. If the bundle generated has a good internal cohesion it is kept, otherwise it is discarded. We call this method `BOBO` and we report its pseudo-code in Algorithm 4.

`BOBO` starts with an empty set of candidate bundles (line 1), and considers each item as a possible pivot (line 2). At each iteration an item is picked from the set $Pivots$. Line 4 determines the way in which pivots are chosen. We can pick pivots uniformly at random among the candidates pivots, or with a certain bias to particular candidate pivots, or do a more sophisticated approach such as picking at each step the furthest pivot to the previous one. Once a pivot is selected we build a bundle S around it. This is done by the routine `pick_bundle` described in Algorithm 5. The routine greedily keeps picking the closest element to the pivot ω (line 3), as far as the complementarity constraint (line 4) and the budget constraint (line 5) are satisfied.

Algorithm 4 `BOBO`

Input: $\mathcal{I}$, α, f, β, minimum bundle score μ, and number of bundles c
Output: a set of c valid candidate bundles

1: $Cand \leftarrow \emptyset$
2: $Pivots \leftarrow \mathcal{I}$
3: **while** $Pivots \neq \emptyset$ **and** $|Cand| < c$ **do**
4: $\omega \leftarrow$ pick an element from $Pivots$
5: $Pivots \leftarrow Pivots \setminus \{\omega\}$
6: $S \leftarrow$ `pickBundle`$(\omega, \mathcal{I}, \alpha, f, \beta)$
7: **if** $score(S) \geq \mu$ **then**
8: $Pivots \leftarrow Pivots \setminus S$
9: $Cand \leftarrow Cand \cup \{S\}$
10: **return** $Cand$

Other greedy objectives for `pick_bundle` can be used. For instance, instead of choosing the element i that maximizes $s(i,\omega)$ one can pick the element that maximizes $s(i,j)$ for $j \in s$, or the one that maximizes $\sum_{j \in s} s(i,j)$. We experimented with these more complex objectives and the results were not substantially better than with the simpler one we report, while the running times were definitively worse.

Let us go back to `BOBO`'s main loop. Once a candidate bundle is created we check (line 7) whether its internal cohesion $score(S) = \sum_{u,v \in S} s(u,v)$ is larger than an input given threshold μ. If also this check is passed then the bundle enters in $Cand$ and its elements are removed from $\mathcal{I}$ and $Pivots$ so that they are no longer used. Instead if the bundle S has a score lower than μ then it is discarded. In both cases the pivot ω is removed from $Pivots$ so that it is no longer considered.

The algorithm might end without having produced c valid bundles due to a too restrictive μ. In this case we might keep the $cand$ set we have produced, if its size is $\gg k$, or we can re-run `BOBO` with a smaller μ.

5 Cluster-and-Pick

We next introduce a totally different method, suggested by the observation that the objective function of composite retrieval has various similarities with a clustering problem.

In a first phase items are clustered based on their compatibilities, to form k clusters with good internal cohesion and external separation. This can be done by means of any standard clustering algorithm. Then in a second phase we pick a good bundle from each cluster (subroutine `BestBundle`, Algorithm 7, which in turn calls subroutine `PickBundle`, Algorithm 5). We refer to this method as `CAP` (Cluster-and-Pick, Algorithm 6). Note that contrary to the algorithms of the previous sections, none of the steps of this approach gives us any kind of approximation guarantee.

Algorithm 5 `pick_bundle`

Input: pivot ω, set of items $\mathcal{I}$, parameters α, f, β
$s \leftarrow \{\omega\}$; $covered \leftarrow \{\omega.C\}$
$active \leftarrow \mathcal{I} \setminus \{\omega\}$; $finish \leftarrow$ false
while not $finish$ **do**
  $i \leftarrow \text{argmax}_{[i \in active]}\ s(i, \omega)$
  if $i.\alpha \notin covered$ **then**
    if $f(s \cup \{i\} \leq \beta)$ **then**
      $s \leftarrow s + i$; $covered \leftarrow covered \cup \{i.\alpha\}$
    else
      $finish \leftarrow$ true
  $active \leftarrow active \setminus \{i\}$
return s

Algorithm 6 `CAP`: Cluster-And-Pick

Input: $\mathcal{I}$, α, f, β,k;
Output: A set $\mathcal{S}$ of k valid bundles.
$clusters \leftarrow clustering(I, k)$
$\mathcal{S} \leftarrow \emptyset$
for each $cluster \in clusters$ **do**
  $\mathcal{S} \leftarrow \mathcal{S} \cup \text{bestBundle}(cluster, \alpha, f, \beta)$;
return $\mathcal{S}$

Algorithm 7 `BestBundle` Routine

Input: set of items $\mathcal{C}$, α, f, β;
Output: one valid bundle
$best \leftarrow 0$
for each $\omega \in \mathcal{C}$ **do**
  $s \leftarrow$ `pickBundle`$(\omega, \mathcal{C}, \alpha, f, \beta)$
  if score$(s) > best$ **then**
    $best \leftarrow s$
return $best$

6 Integer Programming

Like many combinatorial problems, Composite retrieval can be tackled with an integer programming approach. Let us consider the following variables:

$$x_{uj} = \begin{cases} 1, & \text{if } u \in S_j, \\ 0, & \text{otherwise,} \end{cases} \qquad u \in \mathcal{I}, j = 1, \ldots, k$$

$$y_{uvj} = \begin{cases} 1, & \text{if } u \in S_j, \\ & v \in S_j, \\ 0, & \text{otherwise,} \end{cases} \qquad u, v \in \mathcal{I}, u < v$$

$$z_{ij} = \max_{u \in S_i, v \in S_j} s(u, v) \qquad i, j = 1, \ldots, k \quad i < j.$$

Using these definitions, the objective function of our problem may be rewritten as:

$$\gamma \sum_{j=1}^{k} \sum_{\substack{u,v \in S_j \\ u < v}} s(u, v) + (1 - \gamma) \sum_{1 \leq i < j \leq k} \Big(1 - \max_{u \in S_i, \in S_j} s(u, v)\Big)$$
$$= \gamma \sum_{j=1}^{k} \sum_{\substack{u,v \in \mathcal{I} \\ u < v}} s(u, v) y_{uvj} - (1 - \gamma) \sum_{i=1}^{k} \sum_{j=i+1}^{k} z_{ij} + (1 - \gamma) \frac{k(k-1)}{2}.$$

Then the problem can be formulated as maximizing this function subject to:

$$\sum_{u \in \mathcal{I}} c_u x_{uj} \leq \beta \quad j = 1, \ldots, k, \tag{1}$$

$$\sum_{\substack{u \int \mathcal{I} \\ u.\alpha = a}} x_{uj} \leq 1 \quad j = 1, \ldots, k, \quad \forall a \, possible \, value \, of \, \alpha, \tag{2}$$

$$y_{uvj} \leq \frac{x_{uj} + x_{vj}}{2} \quad j = 1, \ldots, k, \qquad u, v \in \mathcal{I}, \tag{3}$$

$$z_{ij} \geq s(u, v)(x_{ui} + x_{vj} - 1), \quad 1 \leq i < j \leq k \, u, v \in \mathcal{I}, \tag{4}$$

$$x_{uj} \in \{0, 1\}, \qquad y_{uvj} \in \{0, 1\}, \qquad 0 \leq z(jj') \leq 1.$$

Constraints (1) assert that the budget does not exceed the threshold for each bundle. Constraints (2) guarantee complementarity (two items with the same value for property α may not be assigned to the same bundle). Constraints (3) ensure that y_{uvj} is equal to 0 if items u, v are not in the same bundle j. Finally, constraints (4) give to z_{ij} the maximum similarity between items in bundles i and j.

For solving the integer programming formulation we implemented a Branch-and-Cut algorithm using CPLEX 12.1. We added to the standard CPLEX algorithm a primal heuristic and valid cutting planes specifically derivated for the problem. The heuristic is a simple fast greedy procedure that is applied on every node of the tree and is capable of finding good solutions in the first stages. The cutting planes, like $\sum_{v \in \mathcal{I} v.\alpha = a} y_{uvj} \leq x_{uj} \forall v \in \mathcal{I}$ and a possible value of α, are very useful to close the initial gap relaxation. These customized components have an important impact on the computational performance of the algorithm, but due to lack of space we cannot extend furtherly on this matter.

7 Experiments

In this section we test the effectiveness and efficiency of our algorithms on two real-world data sets. The first data set corresponds to 10 problem instances of 20 items each. The second data set corresponds to around 150 problem instances of 300-500 items each.

Data set 1 (Attractions in European Cities). A sample of 20 touristic attractions in 10 large European cities (London, Berlin, Madrid, Rome, Paris, Bucharest, Budapest, Hamburg, Vienna, and Warsaw), as listed in their respective pages in Wikipedia. Each of the 200 attractions can belong to one of the three following types: (i) park, zoo, or other open space; (ii) museum or library; (iii) other notable landmark. Each attraction also has a ticket price, obtained by using crowdsourcing via provider Crowdflower,[1] aggregating responses from 60 annotators, with three independent annotators checking on the webpage of each attraction the full ticket price for an adult. The average price was close to €6, with only four attractions costing more than €30. The cost of the 118 free attractions was set to €0.001.

Each query is the name of a city and the candidate items $\mathcal{I}$ are the attractions in that city. For each city, we generate $k = 3$ bundles of recommended attractions. The compatibility between two attractions u, v was computed based on their distance in kilometers $d(u, v)$, approximated using their geographical coordinates as provided in their respective Wikipedia pages. In each city, the maximum distance D was computed and then the compatibility used was $1 - d(u, v)/D + \epsilon$. This means the two attractions that are farther apart have compatibility ϵ. The cost of each city tour is the sum of the costs of each attraction on it, which must not exceed the budget β.

Data set 2 (Restaurants in US Cities). A sample of Yahoo! Local[2] containing user-contributed restaurant reviews. We picked all cities for which there are at least 100 restaurants with reviews. This left us with 38,530 restaurants distributed over 149 US cities, with the largest cities being New York (over 2,000 restaurants) and Los Angeles (over 1,000 restaurants). Most cities in our sample have between 300 and 500 restaurants. Each restaurant has one of the following three average meal prices: cheap ($10), moderate ($20) or expensive ($30). Each restaurant has multiple values for cuisine, drawn from 291 possible cuisine values, including *American*, *Italian*, *Cajun*, *Omelets*, *Contemporary*, *Sandwiches*, etc. The restaurant with the highest number of cuisine types has 34 cuisines ranging from Californian to Spanish.

Each query is the name of a city and the candidate items $\mathcal{I}$ are the restaurants in that city. For each city, we generate $k = 10$ bundles of recommended restaurants. The compatibility between two restaurants u, v is the number of reviewers that have given both restaurants a positive review. The complementarity attribute α is the cuisine type, meaning that on each bundle we do not want two restaurants having a cuisine type in common. The cost of each bundle is the sum of the costs of having a meal at each of the restaurants in the bundle.

Budgets. We set the budget β in three values: small, medium and large. In the case of data set 1 (EU attractions), these correspond to €1, €20, and €50. The small budget basically limits the budget to only the free attractions. In the case of the data set 2 (US restaurants), we used the following three values: $50, $100, and $200. The small budget does not allow many choices in each bundle, while the large budget is often equivalent to an unlimited budget given the distribution of cuisine types and meal prices.

1. http://www.crowdflower.com/.
2. http://local.yahoo.com/.

7.1 Implementation

Bundles one-by-one. We implemented the `BOBO` method described in Algorithm 4, varying the number of bundles that are generated before `ChooseBundles` is invoked to select the k that will be returned. `BOBO-1` is our baseline and it just picks k bundles at random (so `ChooseBundles` becomes trivial). In the case of the large data set, we included `BOBO-5`, that generates $5k$ bundles, `BOBO-10`, that generates $10k$ bundles, and `BOBO-E` (Exhaustive), that picks as pivots all of the items exhaustively and generates $c = |\mathcal{I}|$ candidate bundles with the routine `pick_bundle` described in Algorithm 5. In all but the exhaustive case, an item that has already been picked as a member of a bundle cannot be picked as a pivot, as specified by Algorithm 4. In the case of the small data set, we included `BOBO-E'`, that does discard previously-used bundle members as future pivots, as otherwise bundles tend to be often duplicate. The minimum score of a bundle μ is determined by generating five bundles at random and then picking the median score.

Constrained clustering. The `C-HAC` method described in Algorithm 3 is implemented to stop at k clusters, or when no further merge operations can be executed. Typically, the latter happens much earlier. We experimented with generating more than k clusters and then using `ChooseBundles`; but the results are basically the same as the ones we report here, because anyway the algorithm generates a number of clusters larger than k, typically around $\frac{1}{3}|\mathcal{I}|$.

Cluster-and-pick. The `CAP` method described in Algorithm 6 is implemented using METIS [17] as the clustering method, with k clusters. Procedure `BestBundle` in Algorithm 7 is invoked once for each of the k clusters, returning one bundle per cluster.

Integer programming. We used the branch-and-cut implementation described in Section 6. In the case of the large data set, it is run for a maximum time of 1 minute for each instance.

7.2 Results

Objective function. Experimentally we observe that *the performance of the methods w.r.t. the objective function, highly depends on* γ. Figs. 1 and 2 show the distribution of the value of the objective value of the solutions for $\gamma \in \{0.1, 0.5, 0.9\}$. Results with intermediate values of γ lie between those we show here and are thus omitted. We observe the following:

- The best results are obtained by `IP`. However, due to the high running times, this algorithm may be seen more as a benchmark than as a really feasible approach.
- For small γ, i.e., when inter-bundle separation is important, the clustering-based methods are better than the variants of bundles one-by-one.
- For large γ, i.e., when intra-bundle cohesiveness is important, the BOBO methods are better than the clustering ones.

With the BOBO methods, for the small data set the exhaustive method is better (but comparable) to the `BOBO-1` method. For the large data set the difference is

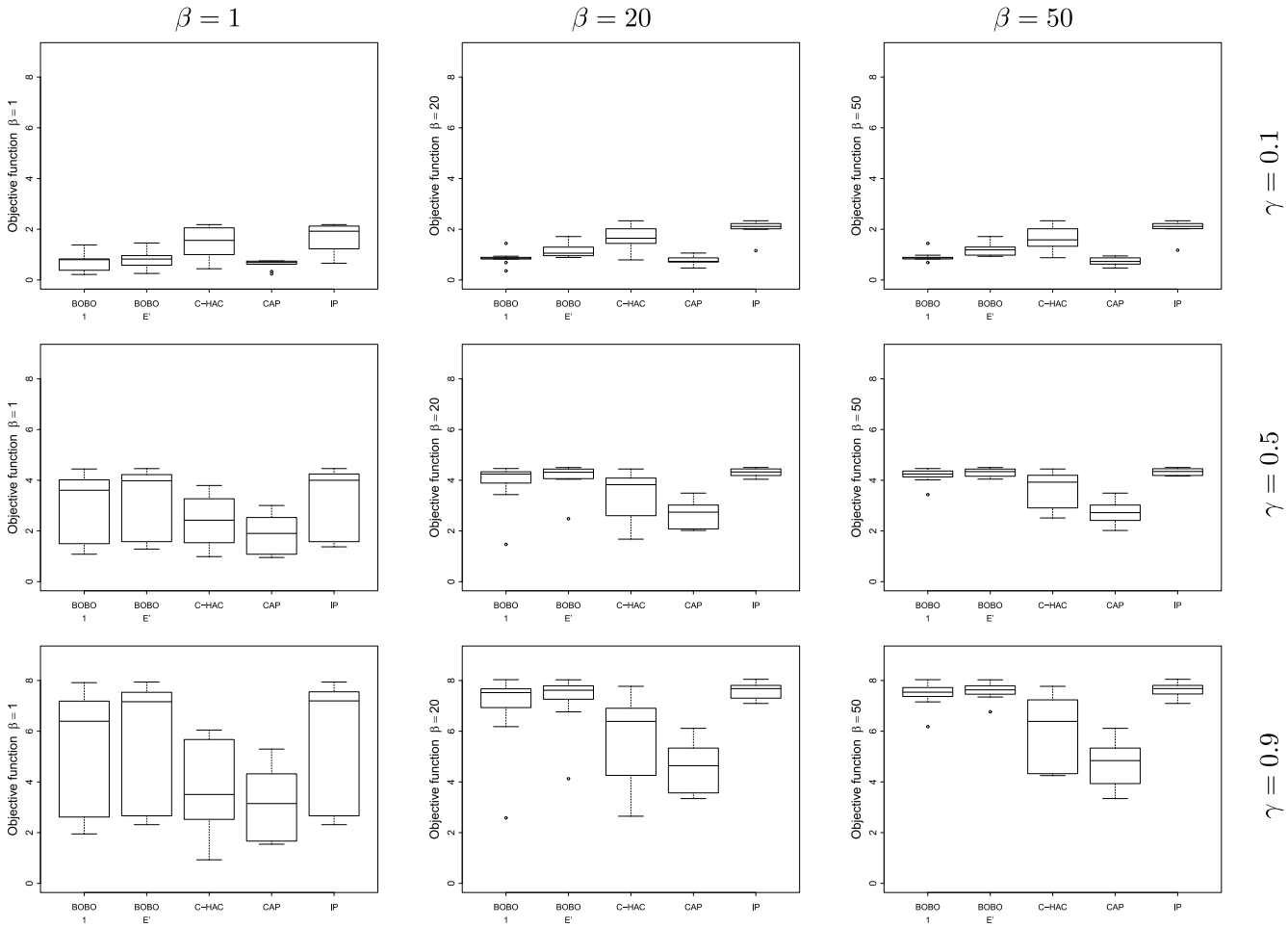

Fig. 1. Results on data set 1 (EU attractions). Objective function for (top to bottom) $\gamma \in \{0.1, 0.5, 0.9\}$ and (left to right) $\beta \in \{1, 20, 50\}$. The scale of the *y*-axis is the same for all plots.

more substantial. Additionally there is a large advantage of BOBO-5 over BOBO-1 in the large data set.

When comparing different clustering methods, in the small data set we see a clear advantage for C-HAC, while on the large data set we see that they are comparable except for cases of large γ and large budget.

Complementarity, score, diversity, and size. To further understand the differences among methods, we can look at some properties of the bundles: number of covered values of the complementarity attribute, bundle score (left-hand side of our objective function in Problem 1), diversity (right-hand side of objective function), and size. A comparison of these quantities is shown in Fig. 3 for $\gamma = 0.5$ (and $\beta = 20$ for data set 1, $\beta = 100$ for data set 2).

The variations among different methods depending on γ can be summarized as follows. Clustering-based methods tend to produce bundles whose elements are not compatible with items from other bundles; this can be seen by the higher diversity of C-HAC in data set 1, and the higher diversity of C-HAC and CAP in data set 2. On the other hand, bundles one-by-one methods tend to produce bundles composed of items that are highly compatible among them, as evidenced by the higher bundle score.

Running time. We ran our experiments on a mid-range Linux server (8×2.6 GHz 64-bit Intel processors). The average and median time (over all problem instances and all parameters settings discussed above) for the different methods is shown on Table 1.

For data set 1, for all methods the mean and median running time across instances were below one second, except for the IP method, which had a mean of 1.94 seconds. For data set 2, the running time of all methods was around 2 to 6 seconds, except for C-HAC, where the average running time is dominated by the larger problem instances. Something similar, but not as pronounced, is observed for BOBO-E in data set 2. The IP method, in the small data set, is on average four times slower, despite many instances in which is quite fast—its median is smaller than those of the other methods in this data set. In the large data set instead, the IP method is one order of magnitude slower than our heuristics. Note that the table excludes 94 problem instances (5.2 percent of them) where the IP formulation was unable to find a solution in 1 minute.

A plot of running time versus instance size for the larger data set (US restaurants) is shown on Fig. 4. The IP method is in general unable to find a solution in under 1 minute for the larger problem instances, and the C-HAC and BOBO-E rapidly increase in running time as problem instances grow (note the logarithmic scale of the *y*-axis). BOBO-1 and CAP can solve larger problem instances faster.

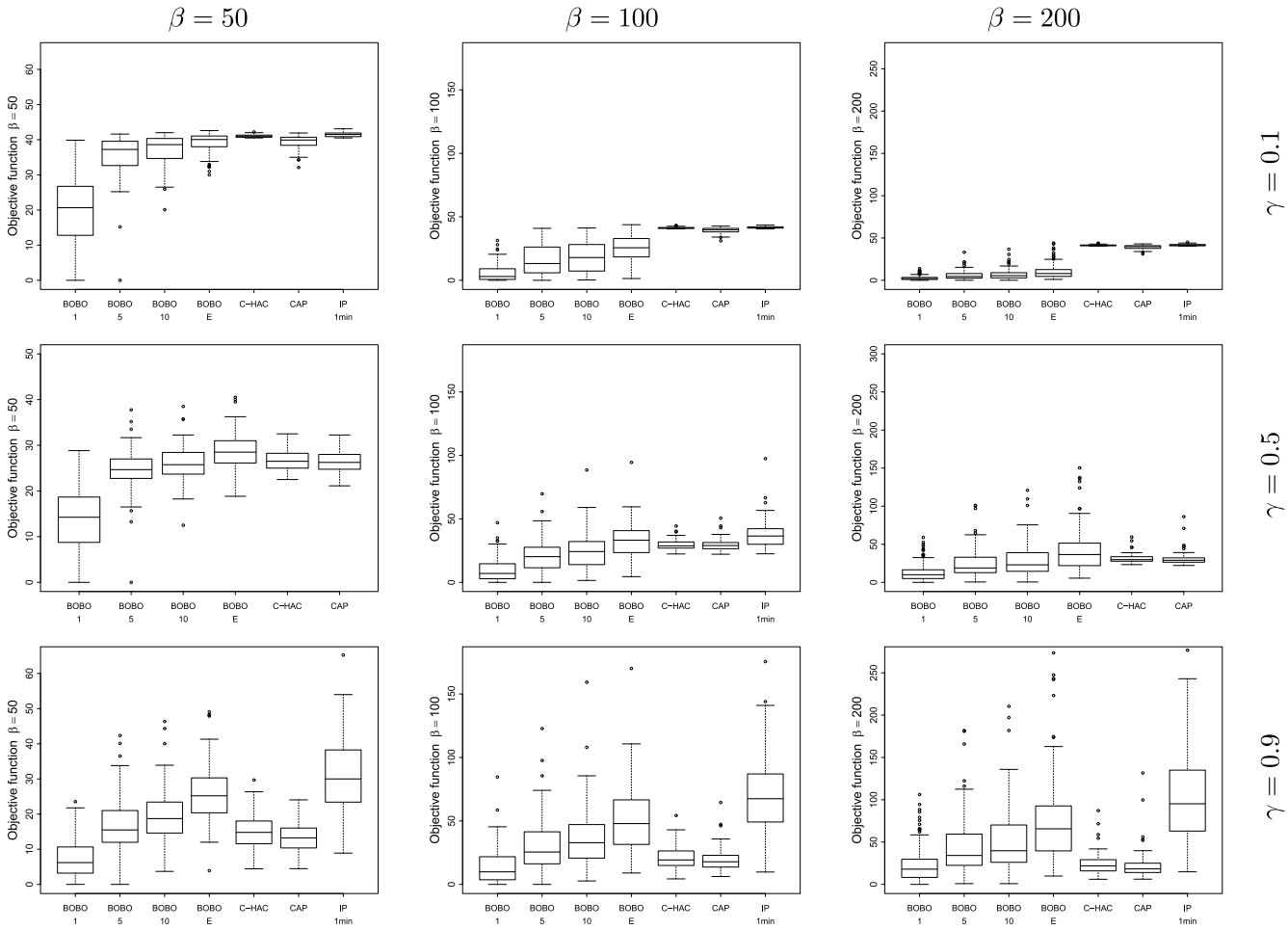

Fig. 2. Results on data set 2 (US restaurants). Objective function for (top to bottom) $\gamma \in \{0.1, 0.5, 0.9\}$ and (left to right) $\beta \in \{50, 100, 200\}$. The scale of the *y*-axis is the same for all plots in the same column.

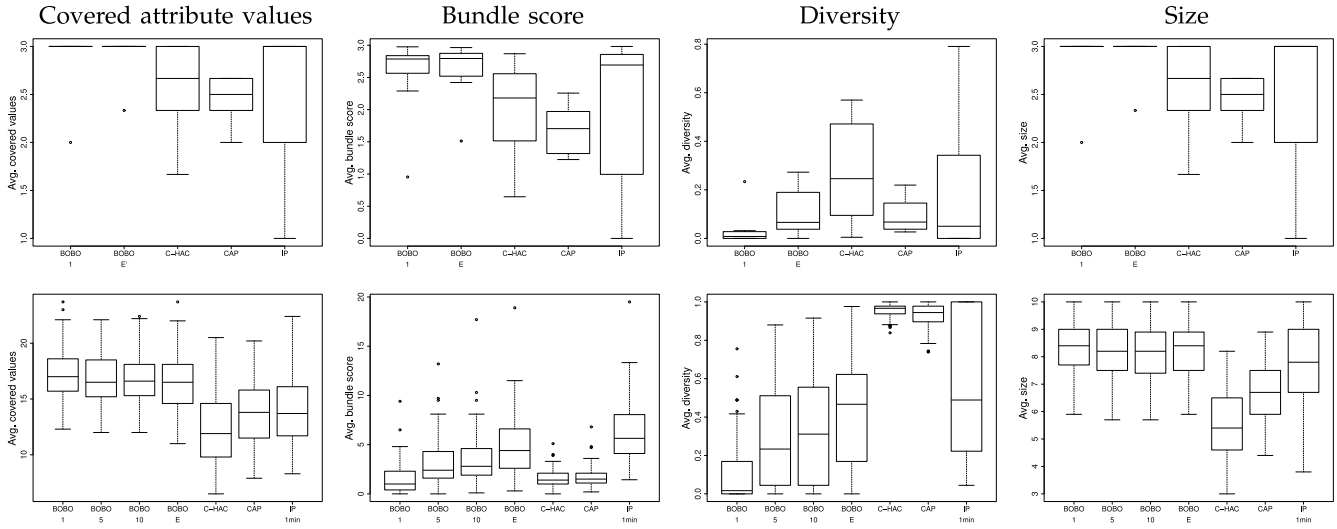

Fig. 3. Average number of covered complementarity attribute values, bundle score, diversity, and size of produced bundles for data set 1 (first row) data set 2 (second row) with $\gamma = 0.5$, $\beta = 100$.

Comparison with the IP *benchmark*. Fig. 5 compares IP versus BOBO–E, the heuristic that performs best in most cases, in the case of data set 2. We use a different color for each $\beta \in \{50, 100, 200\}$. As it may be seen, IP yields a larger value for the objective function in the majority of the cases, specially when the budget is large. In those experiments, IP found the optimal solution in 387 out of 1,815 instances ($\approx$20 percent of the total) with an average running time of 11 seconds, showing a better performance for high budget and for high γ values. However, for 94 instances

TABLE 1
Running Time in Seconds

Method	Dataset 1 Mean	Dataset 1 Median	Dataset 2 Mean	Dataset 2 Median
CAP	0.47	0.48	2.0	2.0
C-HAC	0.45	0.45	19.8	2.9
BOBO 1	0.44	0.44	1.7	1.7
BOBO E	-	-	6.4	2.8
BOBO E′	0.46	0.46	-	-
IP	1.94	0.38	49	60

($\approx$5 percent) it failed to produce any solution, this being more notorious for instances with high budget.

The Choose Phase. To evaluate how important the Choose phase is, we compared the densest-subgraph heuristic `ChooseBundles` in Algorithm 2 with a simpler method, in which we simply pick the top $k = 10$ bundles by score. The results are shown in Fig. 6 and correspond to data set 2. There are significant gains ranging from about 30 percent to almost 300 percent (in terms of the median of the objective function) of using the densest-subgraph heuristic.

Comparison with Xie et al. [20]. Finally, we compared our method with the package-recommendation method described in [20]. This method requires a notion of "importance" for each item, which we create by joining our data set 1 (attractions in European cities) with user ratings in `tripadvisor.com`.

The algorithm of [20] does not attempt to enforce diversity. As a consequence, the bundles it produces naturally tend to lack it. To quantify this, we measured for each pair of bundles generated by a run of the algorithm its Jaccard coefficient J (size of the intersection divided by size of the union), and plot the average of $1 - J$. The mode of this value for [20] is 0.5, which comes from having bundles of three elements that share two of them (the union has four elements and the intersection has two elements, $2/4 = 0.5$). Fig. 7 reports this value of our optimal solution in several instances of the problem, indicating that in almost all cases the method in [20] yields less diversity in the bundle elements than ours. If we measure diversity as the right-hand-side of our objective function, we get values of zero for almost all solutions provided by [20].

If we go for a more qualitative analysis, we find more evidence of the benefit of our approach against those purely

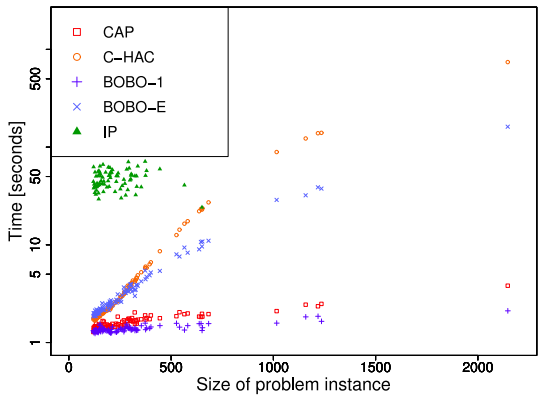

Fig. 4. Average running time for varying instance sizes in data set 2 (US restaurants).

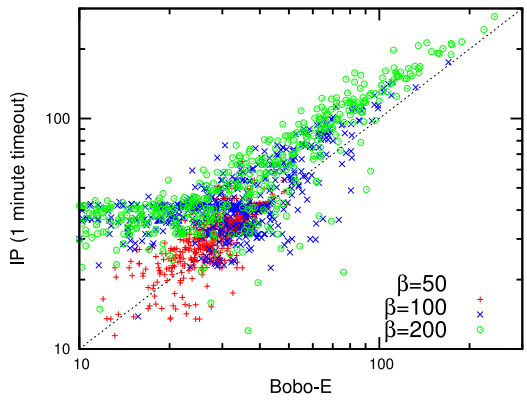

Fig. 5. Objective function for `BOBO-E` versus `IP` in data set 2 (US restaurants). Each dot is a problem instance.

based on the individual quality of the items like [20]. Table 2 shows the optimal solution of our approach when queried for three bundles with different values of diversity factor γ and a budget β =€20 in the city of Rome. We observe many interesting things. The output of [20] consists of three bundles that differ in one attraction, as if the algorithm had chosen the best two attractions in town, and complete with a third one. This is the typical output of that algorithm, with some minor variations depending on the budget. As a

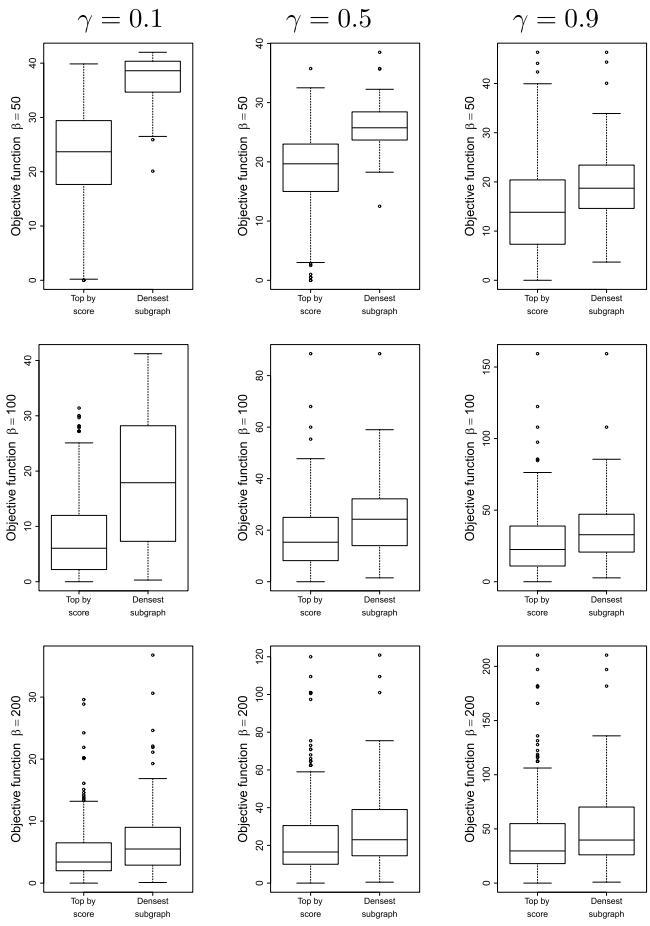

Fig. 6. Comparison of `BOBO-5` (data set 2) using two different methods for the Choose phase: the densest-subgraph heuristic `ChooseBundles`, or simply picking the ones with the highest score.

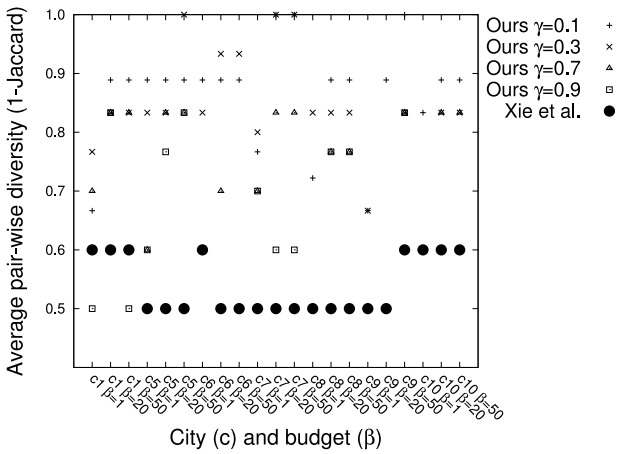

Fig. 7. Average pair-wise score (one minus Jaccard coefficient) computed on data set 1 for different values of γ and β, comparing our algorithm with Xie et al. [20].

consequence of that, it's clear that for a user that has already been to St. Peter's area, the output is almost useless, i.e., there's no useful bundle for his needs. In general, if a user has already been to one of the best attractions in town, no bundle will be good for him. A typical behavior of tourists is to go to the most relevant spots in their first day in the city, so having alternatives not including them for the following days is clearly a benefit. Moreover we observe that many landmarks with rankings almost as high as those that appear in the solution of [20] (like, for example, the Vatican Museums, Villa Giulia or the Colosseum) do not appear at all in that solution. In ours, however, they tend to appear in one or more bundles. In general, if there are many places with high ratings, and ratings are "similar", some very well ranked places disappear completely. Finally, [20] includes in the same bundle places that are very far away from each other, like the Capitoline Museums, St. John Lateran and St. Peter's (second bundle) or Capitoline Museums, Lateran's Palace and St. Peter's (third bundle). Our bundles are rather coherent in that sense. For the sake of justice, [20] proposed to include a distance restriction in their model, which would made disappear that difference between the two algorithms.

8 Related Work

Composite retrieval. The notion of composite retrieval was proposed with different semantics in recent work [3], [7], [9], [18], [20]. CARD [7] is a framework for finding top-k recommendations of packages of products or services. A language similar to SQL is proposed to specify user requirements an the combination of atomic costs. Recommended packages are of fixed size. In [3], the problem is to query documents and build packages formed by multiple entities such as a city, a hotel or an airline. A package in that case is of fixed size (as a simple case of our notion of budget).

In [18], the authors explore retrieving bundles of items in the form of a star (e.g., an iPhone and all its accessories). A bundle is subject to budget and item compatibility constraints (e.g., co-purchasing and co-browsing). In [18], the authors design a graph traversal algorithm that retrieves itineraries in a city. An itinerary is an ordering of a set of points of interest (e.g., landmarks in a city), subject to time constraints. The ordering imposes a different relationship between points of interest in the form of a chain, and makes the retrieval problem significantly different from the model developed in [18]. The solution relies on an adaptation of graph traversal for the orienteering problem. Finally, in [20], the authors explore returning approximate solutions to composite retrieval. The focus of the work is on using a Fagin-style algorithm for variable size bundles and proving its optimality. The same authors, further develop the idea into a prototype of recommender system for travel planning [21].

None of these works accounts for diversity across bundles and formalizes retrieval as a clustering problem that accounts for intra-bundle compatibilities and complementarity as well as a general notion of budget that goes beyond bundle size.

Diversity. Diversifying web search results and recommendations aims to achieve a compromise between relevance and result heterogeneity. In [15], the authors adopt an axiomatic approach to diversity that aims to address user intent. They show that no diversification function can satisfy all axioms together and illustrate that with concrete examples. In [2], taxonomies are used to sample search results in order to reduce homogeneity. In the database context, Chen and Li [8] propose to post-process structured query results, organizing them in a decision tree for easier navigation. In [19], a hierarchical notion of diversity in databases is introduced, and efficient top-k processing algorithms are developed.

In recommendations [23], [11], results are typically post-processed using pairwise item similarity in order to generate a list that achieves a balance between accuracy and diversity. For example, in the recommender systems world, the approach in [23] defines an intra-list similarity which relies on mapping items to taxonomies to determine topics or using item features such as author and genre. The method is based on an exhaustive post-processing algorithm which operates on a top-N list to compute the top-K results ($N > K$). In contrast, in [11], diversity is formulated as a set-coverage problem. These approaches do not retrieve item bundles under validity, diversity and compatibility constraints. In [22] a method based on structural support vector machines is introduced for the learning task of predicting diverse subsets.

Finally, [13] introduces diversity in the framework of sponsored search ads, proposing algorithms for the selection of ads that intend to increase heterogeneity while not significantly reducing revenue and maintaining an

TABLE 2
Bundles Produced by Our Approach and [20] for Different Values of γ and β =€20, in the City of Rome

$\gamma = .1$	Villa_Giulia	Basilica_St._John_Lateran	Vatican_Museum St._Peter's_Basilica St._Peter's_Square
$\gamma = .3$	Vatican_Museum St._Peter's_Basilica St._Peter's_Square	Forum_Romanum Via_Veneto Villa_Borghese_gardens	St._Peter's_Basilica Castel_Sant'Angelo St._Peter's_Square
$\gamma = .5$	Vatican_Museum St._Peter's_Basilica St._Peter's_Square	Forum_Romanum Via_Veneto Villa_Borghese_gardens	St._Peter's_Basilica Castel_Sant'Angelo St._Peter's_Square
[20]	St._Peter's_Basilica Capitoline_Museums St._Peter's_Square	Capitoline_Museums Basilica_St._John_Lateran St._Peter's_Square	Capitoline_Museums Lateran_Palace St._Peter's_Square

incentive for advertisers to keep their bids as high as possible. Heterogeneity is aimed at as a notion that spans various occurrences of the same query, and not just a single one.

A two-pages preliminary version of this work was presented as a poster in [1].

9 Conclusions and Future Work

In many complex search and browsing applications returning item bundles is more appropriate than ranked lists. In this paper, we present composite retrieval as an alternative to ranked lists and formalize the problem as finding the k highest scoring item bundles. Our formulation ensures complementarity among items in the bundles and diversity between retrieved bundles.

In our future work, we plan to investigate variants of the basic definition we provided in this work, e.g., retrieve item bundles that are most compatible with my interests (personalized composite retrieval), find the best item bundles including a specific item, find the best bundle compatible with a given item. These new problems bare similarities with item recommendation that account for a user profile, with the added flexibility of querying those recommendations in a stylized fashion. We conjecture that such queries will simplify retrieval complexity while raising an additional challenge of returning results as fast as possible.

References

[1] S. Amer-Yahia, F. Bonchi, C. Castillo, E. Feuerstein, I. Mndez-Daz, and P. Zabala, "Complexity and Algorithms for Composite Retrieval," *Proc. 22nd Int'l Conf. World Wide Web Companion (WWW)*, pp. 79-80, 2013.

[2] A. Anagnostopoulos, A.Z. Broder, and D. Carmel, "Sampling Search-Engine Results," *World Wide Web*, vol. 9, no. 4, pp. 397-429, 2006.

[3] A. Angel, S. Chaudhuri, G. Das, and N. Koudas, "Ranking Objects Based on Relationships and Fixed Associations," *Proc. 12th Int'l Conf. Extending Database Technology: Advances in Database Technology (EDBT '09)*, 2009.

[4] Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama, "Greedily Finding a Dense Subgraph," *J. Algorithms*, vol. 34, no. 2, pp. 203-221, 2000.

[5] S. Basu, I. Davidson, and K. Wagstaff, *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. First ed., Chapman & Hall/CRC, 2008.

[6] A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, and A. Vijayaraghavan, "Detecting High Log-Densities: An $O(n^{1/4})$ Approximation for Densest k-Subgraph," *Proc. 42nd ACM Symp. Theory of Computing (STOC '10)*, 2010.

[7] A. Brodsky, S.M. Henshaw, and J. Whittle, "CARD: A Decision-Guidance Framework and Application for Recommending Composite Alternatives," *Proc. ACM Conf. Recommender Systems (RecSys '08)*, 2008.

[8] Z. Chen and T. Li, "Addressing Diverse User Preferences in SQL-Query-Result Navigation," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, 2007.

[9] M.D. Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu, "Automatic Construction of Travel Itineraries Using Social Breadcrumbs," *Proc. 21st ACM Conf. Hypertext and Hypermedia (HT '10)*, 2010.

[10] I. Davidson and S.S. Ravi, "Agglomerative Hierarchical Clustering with Constraints: Theoretical and Empirical Results," *Proc. Ninth European Conf. Principles and Practice of Knowledge Discovery in Databases (PKDD)*, 2005.

[11] K. El-Arini, G. Veda, D. Shahaf, and C. Guestrin, "Turning Down the Noise in the Blogosphere," *Proc. 15th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '09)*, 2009.

[12] U. Feige, D. Peleg, and G. Kortsarz, "The Dense k-Subgraph Problem," *Algorithmica*, vol. 29, no. 3, pp. 410-421, 2001.

[13] E. Feuerstein, P.A. Heiber, J. Martínez-Viademonte, and R.A. Baeza-Yates, "New Stochastic Algorithms for Scheduling Ads in Sponsored Search," *Proc. Latin Am. Web Conf. (LA-WEB '07)*, 2007.

[14] A.V. Goldberg, "Finding a Maximum Density Subgraph," technical report, Univ. of California, 1984.

[15] S. Gollapudi and A. Sharma, "An Axiomatic Approach for Result Diversification," *Proc. 18th Int'l Conf. World Wide Web (WWW '09)*, 2009.

[16] R. Hassin, S. Rubinstein, and A. Tamir, "Approximation Algorithms for Maximum Dispersion," *Operations Research Letters*, vol. 21, no. 3, pp. 133-137, 1997.

[17] G. Karypis and V. Kumar, "MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System (version 2.0)," technical report, Dept. of Computer Science, Univ. of Minnesota, 1995.

[18] S.B. Roy, S. Amer-Yahia, A. Chawla, G. Das, and C. Yu, "Constructing and Exploring Composite Items," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, 2010.

[19] E. Vee, J. Shanmugasundaram, and S. Amer-Yahia, "Efficient Computation of Diverse Query Results," *IEEE Data Eng. Bull.*, vol. 32, no. 4, pp. 57-64, Dec. 2009.

[20] M. Xie, L.V.S. Lakshmanan, and P.T. Wood, "Breaking Out of the Box of Recommendations: From Items to Packages," *Proc. Fourth ACM Conf. Recommender Systems (RecSys '10)*, 2010.

[21] M. Xie, L.V.S. Lakshmanan, and P.T. Wood, "Comprec-Trip: A Composite Recommendation System for Travel Planning," *Proc. IEEE 27th Int'l Conf. Data Eng. (ICDE '11)*, 2011.

[22] Y. Yue and T. Joachims, "Predicting Diverse Subsets Using Structural Svms," *ICML*, 2008.

[23] C.-N. Ziegler, S.M. McNee, J.A. Konstan, and G. Lausen, "Improving Recommendation Lists Through Topic Diversification," *Proc. 14th Intl. Conf. World Wide Web (WWW '05)*, 2005.

Sihem Amer-Yahia is currently the director of research (DR1) at the National Center for Scientific Research (CNRS) in the Grenoble Informatics Laboratory (LIG). Her research interests include the intersection of large-scale data management and analytics with an application to the social web. She is a member of the Very Large Data Base (VLDB) Endowment and serves on the editorial boards of the *ACM Transactions on Database Systems* (TODS), the *VLDB Journal*, and the *Information Systems Journal*.

Francesco Bonchi is currently a senior research scientist at Yahoo! Research and the head of the Web Mining Research Group in Barcelona, Spain. His current research interests include mining query-logs, social networks, and social media. He is a member of the editorial boards of the *IEEE Transactions on Knowledge and Data Engineering* (TKDE) and the *ACM Transactions on Intelligent Systems and Technology* (TIST). More information can be found at http://www.francescobonchi.com/.

Carlos Castillo is currently a senior scientist at the Qatar Computing Research Institute in Doha, where he is in the mining of content, links, and usage data from the web. After being a research associate at the Sapienza University of Rome and Universitat Pompeu Fabra, he was with Yahoo! Research for six years . More information can be found at http://www.chato.cl/research/.

Esteban Feuerstein is an associate professor in the Department of Computer Science, FCEyN, the University of Buenos Aires, Argentina. His research focuses in algorithms and data structures, online, parallel and distributed problems and algorithms, with applications in knowledge management, information retrieval, microeconomy, and algorithmics of online advertising.

Isabel Mendez-Diaz is an associate professor in the Department of Computer Science, FCEyN, Buenos Aires University, Argentina. Her research interests include mainly the area of combinatorial optimization and integer programming algorithms with applications on graph coloring, vehicle routing, and scheduling.

Paula Zabala is a professor in the Department of Computer Science of the University of Buenos Aires, Argentina, since 2008. Her research interests include combinatorial optimization and integer programming. She has worked on optimization problems in vehicle routing, graph coloring, revenue management, and network design.